PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-269912
(43)Date of publication of application : 14.10.1997

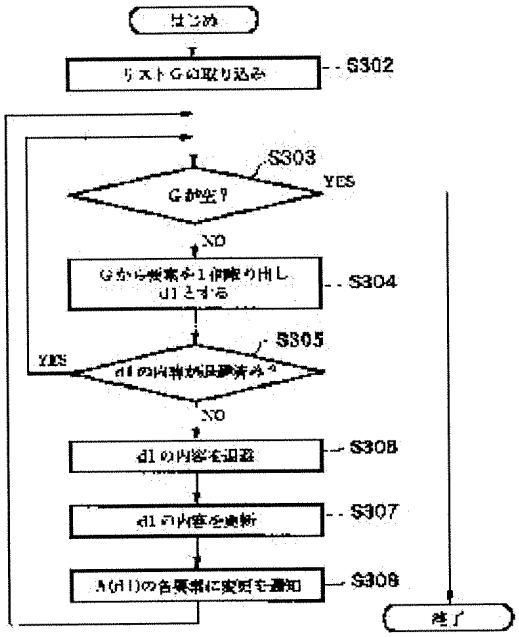
(51)Int.Cl. G06F 12/00
G06F 12/00

(21)Application number : 08-080179
(22)Date of filing : 02.04.1996

(71)Applicant : CANON INC
(72)Inventor : YOSHIMOTO MASAHIKO

(54) INFORMATION PROCESSING METHOD AND INFORMATION PROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To provide an information processing method and an information processor, which can facilitate an interactive processing between users at the time of referring to common data between the plural application software.
SOLUTION: This information processing method is to control the reference of common data by the plural applications. Correction/addition data against common data based on the request of a first application is stored. Common data is updated (S305) based on correction/addition data stored based on the committing request of the first application. Then, information on updated common data is informed to the application referring to other common data (S306).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(2)

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平9－269912

(43)公開日　平成９年(1997)10月14日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　12/00	５３３		Ｇ０６Ｆ　12/00	５３３Ｆ	
	５１８			５１８Ａ	

審査請求　未請求　請求項の数８　ＯＬ　（全　８　頁）

(21)出願番号　特願平８－80179

(22)出願日　平成８年(1996)４月２日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　吉本　雅彦
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

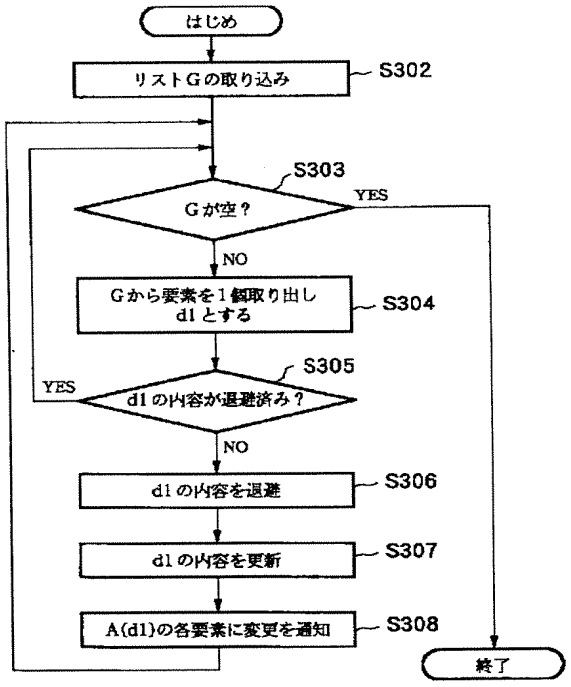
(74)代理人　弁理士　大塚　康徳　(外１名)

(54)【発明の名称】　情報処理方法とその装置

(57)【要約】

【課題】　複数のアプリケーションソフトウエア間で共有データを参照する場合、ユーザ間の対話的処理を容易に実現できる情報処理方法とその装置を提供する。

【解決手段】　複数のアプリケーションが共有データを参照する制御を行う情報処理方法であって、第１のアプリケーションの要求に基づく共有データに対する修正／追加データを格納する。そして、第１のアプリケーションのコミット要求に基づいて、格納された修正／追加データに基づいて、前記共有データを更新する（Ｓ３０５）。そして、更新された共有データに関する情報を他の前記共有データを参照中のアプリケーションに通知する（Ｓ３０６）。



1

【特許請求の範囲】

【請求項１】　複数のアプリケーションが共有データを参照する制御を行う情報処理方法であって、
第１のアプリケーションの要求に基づく共有データに対する修正／追加データを格納する格納工程と、
前記第１のアプリケーションのコミット要求に基づいて、前記格納工程で格納された修正／追加データに基づいて、前記共有データを更新する更新工程と、
前記更新工程で更新された共有データに関する情報を他の前記共有データを参照中のアプリケーションに通知する通知工程とを備えることを特徴とする情報処理方法。

【請求項２】　前記第１のアプリケーションのアボード要求に基づいて、前記更新工程で更新された共有データを、前記格納工程で格納される以前のデータに復帰させる復帰工程をさらに備えることを特徴とする請求項１に記載の情報処理方法。

【請求項３】　前記共有データをアクセスするアプリケーションを検知する検知工程と、
前記検知工程で検知されたアプリケーションがアクセスするデータに対して、当該アプリケーションが共有ロックまたは排他ロックを確保していない場合、当該アプリケーションが前記更新工程を行うに先だって、共有ロックを設定する設定工程とをさらに備えることを特徴とする請求項１に記載の情報処理方法。

【請求項４】　前記復帰工程はさらに、前記更新工程で更新された共有データを、前記格納工程で格納される以前のデータに復帰させる旨の情報を、前記共有データを使用中の他のアプリケーションに通知する復帰通知工程を備えることを特徴とする請求項２に記載の情報処理方法。

【請求項５】　複数のアプリケーションが共有データを参照する制御を行う情報処理装置であって、
第１のアプリケーションの要求に基づく共有データに対する修正／追加データを格納する格納手段と、
前記第１のアプリケーションのコミット要求に基づいて、前記格納手段で格納された修正／追加データに基づいて、前記共有データを更新する更新手段と、
前記更新手段で更新された共有データに関する情報を他の前記共有データを参照中のアプリケーションに通知する通知手段とを備えることを特徴とする情報処理装置。

【請求項６】　前記第１のアプリケーションのアボード要求に基づいて、前記更新手段で更新された共有データを、前記格納手段で格納される以前のデータに復帰させる復帰手段をさらに備えることを特徴とする請求項５に記載の情報処理装置。

【請求項７】　前記共有データをアクセスするアプリケーションを検知する検知手段と、
前記検知手段で検知されたアプリケーションがアクセスするデータに対して、当該アプリケーションが共有ロックまたは排他ロックを確保していない場合、当該アプリケーションが前記更新工程を行うに先だって、共有ロックを設定する設定手段とをさらに備えることを特徴とする請求項５に記載の情報処理装置。

【請求項８】　前記復帰手段はさらに、前記更新手段で更新された共有データを、前記格納手段で格納される以前のデータに復帰させる旨の情報を、前記共有データを使用中の他のアプリケーションに通知する復帰通知手段を備えることを特徴とする請求項７に記載の情報処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、情報処理方法とその装置、特に、トランザクション間の並行性を向上させる情報処理方法とその装置に関する。

【０００２】

【従来の技術】従来より、データベースシステムにおけるトランザクション処理においては、ロック機構を用いた並行制御が用いられてきた。このロックの種類としては一般に、データの更新を行う際に必要となる排他ロックと、データの参照を行う際に必要となる共有ロックがある。

【０００３】さらにロックの獲得と解放は、２フェーズロッキングと呼ばれる方式、すなわち、トランザクション間の相互干渉を防止し、結果として、一貫性のあるスケジューリングを保証する方式に従って行われるのが通例であった。

【０００４】

【発明が解決しようとする課題】しかしながら、従来の方式では、対話的な操作環境を有するアプリケーションには適していないという問題点があった。例えば、あるユーザの操作により、データの更新を目的としてトランザクションが起動されたとき、該データに対しては排他ロックの獲得が必要である。しかし、そのためには、他のトランザクションによって、排他、もしくは共有ロックの確保が行なわれていないことが必要となる。

【０００５】これによれば、以下のような問題点が生じる。すなわち、上記データの内容が、例えば、ウインドウシステム上のウインドウ内に表示されている場合、ウインドウの内容の一貫性を保証するためには、共有ロックの獲得が必要であるが、これによれば、他のトランザクションによる排他ロックの獲得は不可能であり、ウインドウに表示されている間のデータの更新を行うことができなくなる。

【０００６】一方、一貫性の保証を放棄して、共有ロックの確保に続いてデータの読出しを行い、該データの内容をアプリケーション内のメモリ領域に複写したのち、該獲得した共有ロックを解放すれば、他のトランザクションによるデータの更新は可能になる。しかし、この場合であっても、排他ロックが確保されている間は、後続するトランザクションによる共有ロックの確保は不可能

である。この結果、排他ロックが確保される以前に読み出しを終えたアプリケーションにおいては、必ずしも一貫性の保証されないデータが保持され、他方、排他ロックが確保された以後に読み出しを試みようとするアプリケーションにおいては、排他ロックが解放されるまでの間、予測できない待ち時間が生じることになる。これらの振舞いは、複数のユーザが協調しながら、データを共有する分散環境においては、著しく不適当である。本発明は、以上説明した問題点、すなわち、従来のトランザクション処理、特に並行制御機構を用いて、複数ユーザ間で共有されるデータを用いた対話的処理の実現が困難であったという問題点を解決する。

【０００７】本発明は、上記従来例に鑑みてなされたもので、複数のアプリケーションソフトウエア間で共有データを参照する場合、従来のトランザクション処理機構にとっては、対話的な処理の実現が困難であったという問題点を解決した情報処理方法とその装置を提供することを目的とする。

【０００８】

【課題を解決するための手段】上記目的を達成するため、本発明の情報処理方法とその装置は以下の構成を備える。即ち、複数のアプリケーションが共有データを参照する制御を行う情報処理方法であって、第１のアプリケーションの要求に基づく共有データに対する修正／追加データを格納する格納工程と、前記第１のアプリケーションのコミット要求に基づいて、前記格納工程で格納された修正／追加データに基づいて、前記共有データを更新する更新工程と、前記更新工程で更新された共有データに関する情報を他の前記共有データを参照中のアプリケーションに通知する通知工程とを備える。

【０００９】また、別の発明は、複数のアプリケーションが共有データを参照する制御を行う情報処理装置であって、第１のアプリケーションの要求に基づく共有データに対する修正／追加データを格納する格納手段と、前記第１のアプリケーションのコミット要求に基づいて、前記格納手段で格納された修正／追加データに基づいて、前記共有データを更新する更新手段と、前記更新手段で更新された共有データに関する情報を他の前記共有データを参照中のアプリケーションに通知する通知手段とを備える。

【００１０】

【発明の実施の形態】はじめに、本発明の実施の形態の情報処理方法とその装置のポイントを要約した後に、その詳細な説明に入るものとする。本発明の実施の形態の情報処理方法のポイントは、データベースの所望のデータ領域（Ｄ１）に対して共有ロックをかけずにアクセスしているアプリケーションソフトウエア（ＡＰ１）に対して、別のアプリケーションソフトウエア（ＡＰ２）が上述のデータ領域（Ｄ１）を更新するトランザクションの途中で、更新内容をコミットした時に、その更新が発生したことを通知する。そして、別のアプリケーションソフトウエア（ＡＰ２）が、その後、アボードした場合、即ち、更新を無効として、データ領域（Ｄ１）の元の内容に戻す処理を行う。そして、その元の内容に戻した処理を行ったことを、再び、アプリケーションソフトウエア（ＡＰ１）に対して通知する。この様な処理構成を備えることによって、複数ユーザ間で共有されるデータを用いた対話的処理を容易に行うことができる。

【００１１】尚、本処理構成では、上述のアプリケーションソフトウエア（ＡＰ１）に対しては、アプリケーションソフトウエア（ＡＰ１）が参照したいデータ領域に対する共有ロックが係っているか否かにかかわらず、データ参照操作を許可する。

［実施例１］以下、本発明の実施の形態の情報処理方法とその装置について、図を用いて説明する。なお、本発明を実施する装置としては、周知のワークステーション、もしくは、パーソナルコンピュータでよい。

【００１２】図１は、所定のアプリケーションソフトウエアからのデータ参照要求を処理する手順を示すフローチャートである。ここで、このアプリケーションソフトウエアの識別子を”ａ”，データの識別子を”ｄ”とする。また、図２は、図１のフローチャートが示す処理の基本要素を記述した図である。図２は、アプリケーションソフトウエアａ（１０００）、アプリケーションソフトウエアｂ（１００１）、...、アプリケーションソフトウエアｍ（１００２）が本情報処理装置内に存在している例を示している。また、各アプリケーションソフトウエアが、ＤＢ（データベース）（３０００）中のデータ領域”ｄ”を共通にアクセスする場合を示している。

【００１３】アプリケーションソフトウエアａ（１０００）に対応する参照データ識別子リストＤ（ａ）（２０００）は、アプリケーションソフトウエアａ（１０００）がＤＢ（３０００）のどのデータをアクセス中であるかを示すためのもので、参照中の各データの識別子を格納したリストである。データ参照アプリケーションリストＡ（ｄ）（２００１）は、ＤＢ（３０００）中のデータ”ｄ”をアクセス中の各アプリケーションソフトウエアの識別子を格納したリストである。

【００１４】これらの参照データ識別子リストＤ（ａ）（２０００）とデータ参照アプリケーションリストＡ（ｄ）（２００１）は、図７を参照して後述するメモリ２０２の一部の領域にアサインされている。以下、図２を図１のフローチャートの示す処理手順の理解を容易にするための補助図として参照しながら、アプリケーションソフトウエアからのデータ参照要求を処理する手順を説明する。この例では、排他ロックや共有ロックをかけずにデータベースをアクセスする処理に関して説明する。図１のフローチャートの示す処理は、所定のトランザクションマネージャ（ＴＭ）などによって、実行される。

【００１５】まず、ステップＳ１０２では、アプリケー

ションソフトウエアａから、ＴＭに対して、データベースのデータｄを参照する要求があると、ＴＭは、アプリケーションソフトウエアａが参照しているデータ識別子のリストＤ（ａ）を取り出す。ステップＳ１０３では、Ｄ（ａ）にｄが含まれているかどうか判定する。そして、含まれていれば、アプリケーションソフトウエアａは、既にデータｄの参照処理を起動したものとみなして、直ちに処理を終了する。逆に、含まれていなければ、次のステップＳ１０４に移行する。

【００１６】ステップＳ１０４では、ｄを参照しているアプリケーションソフトウエアのリストＡ（ｄ）を取り出す。ステップＳ１０５では、Ａ（ｄ）にはａは含まれていないので、Ａ（ｄ）にａを追加する。ステップＳ１０６では、識別子ｄの指すデータ内容をアプリケーションソフトウエアａに転送する。

【００１７】ステップＳ１０７では、Ｄ（ａ）にｄを追加して、処理を終了する。以上、アプリケーションソフトウエアａからデータベースに対するデータ参照要求を処理する手順を説明した。尚、図１において、２つのリストＤ（ａ）とＡ（ｄ）との一貫性は、公知の手法によって常に保たれているものとする。

【００１８】また、ステップＳ１０２～Ｓ１０４、および、ステップＳ１０５～Ｓ１０７の処理順序は、実施しているデータ共有機構の実装形態に依存するものであり、本発明は、この処理順序に制限されるものではない。図１に示した手順の逆操作、すなわち、データ参照の解除については、図１と全く同様である。すなわち、アプリケーションソフトウエアａとデータｄについて、Ｄ（ａ）にｄが含まれるとき、Ｄ（ａ）からｄを削除するとともに、Ａ（ｄ）からａを削除すればよい。

【００１９】図３は、ＴＭが、図１の手順に従って共有データに対する参照処理を開始した後に、各アプリケーションからアクセスされるデータベースのデータに対する共有ロックの設定処理の説明を行う。図４は、共有ロックの設定対象を管理するリストＬ，Ｌ’の構成例を示す図である。

【００２０】図４の共有ロック要求識別子リストＬ（４０００）は、ＴＭが生成したもので、各アプリケーションソフトウエアからの共有ロックの獲得処理を要求するデータ識別子のリストである。これは、ＴＭが、データベースへの各アプリケーションからのアクセスを監視して生成した、データベースへアクセスのあったデータのリスト、即ち、上述した

Ｄ（ｘ）；ｘ＝ａ，ｂ，．．．ｃ

に等価のリストであるか、または、Ｄ（ｘ）のうち、アプリケーションソフトウエアから共有ロック要求が行われたデータのリストである。

【００２１】ここで、この共有ロック要求識別子リストＬ（４０００）は、図７を参照して後述するメモリ２０２の一部の領域にアサインされている。図３のステップＳ２０２では、上述の共有ロック要求識別子リストＬ（４０００）の要素であって、既に、共有ロック、または、排他ロックが確保されているものを取り除いたリストＬ’（非ロックデータ識別子リスト）を生成する。

【００２２】ステップＳ２０３では、リストＬ’が空かどうかをチェックする。そして、空でなければ、ステップＳ２０４へ進む。空であれば、処理を終了する。ステップＳ２０４では、リストＬ’の各要素に対する共有ロックの確保を行い、処理を終了する。以上の処理をＴＭが繰り返すことによって、共有ロックの必要なデータに対して、適宜、共有ロックをかけていくことができる。

【００２３】ＴＭは、また上述の処理と共に、データベースへの各アプリケーションからのアクセスの内のデータ更新の発生を監視して、データ更新の発生したデータの識別子を羅列したデータリストＧを生成する。次に、図５は、各アプリケーションがデータ更新（コミット）を行うトランザクションにおけるＴＭの処理手順を示すフローチャートである。

【００２４】まず、ステップＳ３０２では、リストＧのデータを取り込む。ステップＳ３０３では、リストＧが空かどうかチェックして、空であれば処理を終了する。他方、空でなければステップＳ３０４へ進む。ステップＳ３０４では、リストＧから要素を１つ取り出してｄ１とする。ステップＳ３０５では、ｄ１の内容が既に退避済みかどうかチェックする。そして、退避済みであれば、ステップＳ３０３に戻る。逆に、退避済みでなければ、ステップＳ３０６に進む。

【００２５】ステップＳ３０６では、ｄ１の内容を一時的に退避する。ステップＳ３０７では、ｄ１の内容を更新する。ステップＳ３０８では、ｄ１の参照を行っているリストＡ（ｄ１）の各アプリケーションソフトウエアに対し変更の通知を行う。そして、ステップＳ３０３に戻り、同様の処理を繰り返す。

【００２６】以上の処理によって、所定のアプリケーションがデータ更新した情報を、参照中の他のアプリケーションに知らせることができる。尚、ステップＳ３０６におけるｄ１の内容の退避領域は、最終的にトランザクションが終了した時点で解放される。ここで、トランザクションが最終的に終了するとは、コミットまたはアボートのいずれかである。ここで、コミットとは、更新した内容を確定する処理である。また、アボートとは、更新した内容を確定せずに廃棄する、言い換えれば、更新前の状態に戻す処理を意味する。

【００２７】従って、コミットであった場合には、単に、退避領域が解放される。一方、アボートであった場合には、ｄ１の退避領域が空でなければ、一時的に行われた更新処理の内容が破棄され、ｄ１の内容として退避領域の内容が再設定される。この時、Ａ（ｄ１）が空でなければ、ステップＳ３０８と同様にＡ（ｄ１）の各要素に対して変更の通知が行われる。

【００２８】以上、説明した処理手順を用いれば、例えば、あるアプリケーションソフトウエアａ１とａ２において、効果的なデータ共有を行うことができる。これを図６を参照して説明する。図６に示すように、データベースにデータｄｔ１，ｄｔ２，ｄｔ３，ｄｔ４があるとし、アプリケーションソフトウエアａ１とａ２がそれぞれ図１の手順に従って、ｄｔ１，ｄｔ２，ｄｔ３、および、ｄｔ１，ｄｔ２，ｄｔ４の参照手順を開始したとする。

【００２９】ここで、アプリケーションソフトウエアａ１とａ２が、それぞれｄｔ１とｄｔ２の更新を行うためにトランザクションを開始したとする。このとき、図３に示した処理手順において、共有ロックを確保するデータ識別子のリストは、アプリケーションソフトウエアａ１についてはｄｔ３であり、ａ２についてはｄｔ４を含むものとする。

【００３０】図３に示した手順によって上記の共有ロックが確保されれば、何らかの時点でａ１とａ２はそれぞれｄｔ１とｄｔ２に関する排他ロックを確保することになるが、ａ１とａ２間のロックの競合が存在しないため、他の外乱要因がなければこれらはともに成功する。この段階で、図５で示した手順に従って一時的な内容更新を行えば、アプリケーションソフトウエアａ１とａ２は、それぞれ、他方が行った更新が直ちに通知され、それに従って処理を行うことが可能である。

【００３１】なお、一般には、データ更新トランザクションの途中で、他のトランザクションによる一時的な更新通知が行われるのは好ましくない。これは、前者のトランザクションにおける一時的な更新内容の評価によっては、最終的に、コミットを行う段階で一貫性が保証されない可能性があるためである。従って、通常のトランザクション開始時には、図１の手順で参照を開始した全ての共有データについて図３に示した手順を適用するのが適当である。

【００３２】さらに、トランザクションとしては、データの更新を行わない類のトランザクションもあるが、これについても図３に示した手順を適用することは可能である。　この場合のトランザクションの目的は、トランザクションの間に他のトランザクションに更新を禁止することである。本実施の形態においては、データ参照処理における共有ロックの確保は不用であって、むしろ不用なロックによりトランザクション間の競合を回避するために、読み出しトランザクションは必要最小限にとどめるのが適当である。

［実施の形態２］以上説明した実施の形態においては、トランザクションにおけるロックの獲得・解放処理を２フェーズで行うものとしていた。

【００３３】しかしながら、本実施の形態に係る更新処理と参照処理が共存し、なおかつ、一時的な更新処理を支援するようなシステムにおいては、いわゆる「serializability」を犠牲にしても、データの共有を許すのが適当である。これを実現するために、明示的なロック解放命令をアプリケーションに提供する。図７は、明示的なロック解放命令を受信した時のＴＭの処理手順を示すフローチャートである。

【００３４】尚、本フローチャートにおいて、排他ロックを解放して共有ロックを確保する手順を示すが、排他ロックの解放処理や共有ロックの解放処理についても同様である。ただし、共有ロックの解放処理については、後述する一時的な更新処理は行われない。ステップＳ４０１では、明示的にロックの解放を行うデータ識別子ｄ２を受信する。

【００３５】ステップＳ４０２では、ｄ２に関する排他ロックが既に確保されているかどうかをチェックする。そして、排他ロックの確保が行われていなければ処理を終了する。さもなければ、ステップＳ４０３へ進む。ステップＳ４０３では、ｄ２の内容を変更したかどうかを判定する。そして、変更を行っている場合には、ステップＳ４０４に進む。変更がなければ、ステップＳ４０５に進む。

【００３６】ステップＳ４０４では、図５に示した手順に従って一時的に更新内容を反映させる。ステップＳ４０５では、排他ロックを共有ロックに変更して、処理を終了する。　尚、ステップＳ４０５における処理は不可分に行われるものとし、かつ、該ロックの解放を待っている他のトランザクションに優先して行われるものとする。

【００３７】図７においては、２フェーズロックに従わない個々のトランザクションについて処理手順を示した。しかし一般には、入れ子になったトランザクション処理が行われることがあり、上位のトランザクションが２フェーズロックに従うにもかかわらず、下位のサブトランザクションにおいて図７の手順を行うと矛盾が生じる。

【００３８】これを防ぐために、上位トランザクションのいずれかがロックの獲得・解放処理を２フェーズロックに従う場合には、サブトランザクションにおけるロックの獲得・解放処理は指定かかわらず２フェーズで行うものとする。この場合、上記明示的なロック解放命令は遅延評価され、最終的には２フェーズロックに従わない上位トランザクションに戻った時点で評価される。

【００３９】次に、図８は、上述した各フローチャートの処理を実行する情報処理装置の構成例を示す図である。この図で、ＣＰＵ２００は、本情報処理装置全体の制御を、メモリ２０２に格納された各種プログラムを読み出し、解釈し、実行することで行う。メモリ２０２には、上述した各フローチャートに対応するプログラムが格納され、また、データベースが格納されている。

【００４０】キーボード２０３とポインティングデバイス２０４は、コマンドやデータの入力を行う。ディスプ

レイモニタ２０１は、ＣＰＵ２００での処理結果や、キーボード２０３とポインティングデバイス２０４から入力したコマンドやデータを表示する。なお、本発明は、複数の機器から構成されるシステムに適用しても、一つの機器からなる装置に適用してもよい。

【００４１】また、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはＣＰＵやＭＰＵ）が記憶媒体に格納されたプログラムコードを読出し実行することによっても、達成されることは言うまでもない。

【００４２】この場合、記憶媒体から読出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。プログラムコードを供給するための記憶媒体としては、例えば、フロッピディスク，ハードディスク，光ディスク，光磁気ディスク，ＣＤ－ＲＯＭ，ＣＤ－Ｒ，磁気テープ，不揮発性のメモリカード，ＲＯＭなどを用いることができる。

【００４３】また、コンピュータが読出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているＯＳ（オペレーティングシステム）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【００４４】さらに、記憶媒体から読出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【００４５】本発明を上記記憶媒体に適用する場合、その記憶媒体には、先に説明したフローチャートに対応するプログラムコードを格納することになるが、簡単に説明すると、上述の各フローチャートに対応する各プログラムモジュールを記憶媒体に格納することになる。以上説明したように、本実施の形態の情報処理方法では、アプリケーションソフトウエアに対して、共有ロックの獲得の有無にかかわらずデータ参照操作を許可する。また、該データ参照操作を行っているアプリケーションソフトウエアがデータ更新処理を目的とするトランザクションを開始する際に、そのトランザクションの起動に先だって、上記データ参照操作が起動されている一連の共有データに対して共有ロックを確保するかどうかを指定する。そして、該データ更新処理を目的とするトランザクションの処理の途中で一時的に更新内容をコミットした場合、そのコミット処理でデータを変更したことを知らせる変更通知を、上記データ参照操作を起動している一連のアプリケーションソフトウエアに対して通知する。

【００４６】また、上記トランザクションが最終的にアボートされた場合には、上記コミット処理に対応する逆操作を行って、元のデータを回復する。そして、その逆操作における変更通知を、上記データ参照操作を起動している一連のアプリケーションソフトウエアに対して通知する。尚、上記データ更新処理を目的とするトランザクションは、ロックの獲得・解放処理を２フェーズロックに従って行うかどうかを指定することができる。

【００４７】以上の基本処理構成によって、複数ユーザ間で共有されるデータを用いた対話的処理を容易に実現することができる。

【００４８】

【発明の効果】以上説明したように本発明によれば、複数のアプリケーションソフトウエア間で共通データを参照する場合、従来のトランザクション処理機構にとっては、対話的な処理の実現が困難であったという問題点を解決できる。

【図面の簡単な説明】

【図１】本発明に係る一実施の形態の各アプリケーションからデータベースの所定のデータに対するデータ参照要求を処理する手順を示すフローチャートである。

【図２】図１の処理を説明するための図である。

【図３】共有ロックの獲得処理手順を示すフローチャートである。

【図４】共有ロック要求データ識別子リストと非ロックデータ識別子リストの構成例を示す図である。

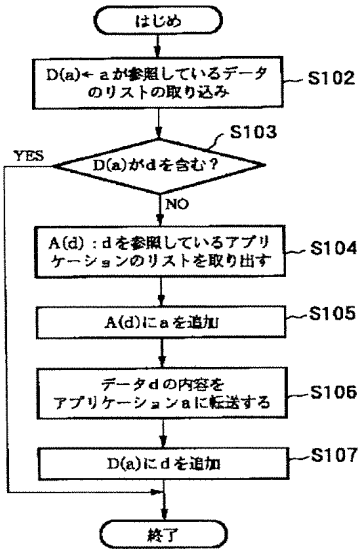
【図５】トランザクションの途中で一時的にデータ更新内容を、所定のアプリケーションに通知する処理する処理手順を示すフローチャートである。

【図６】第１の実施の形態の処理例を説明するための図である。

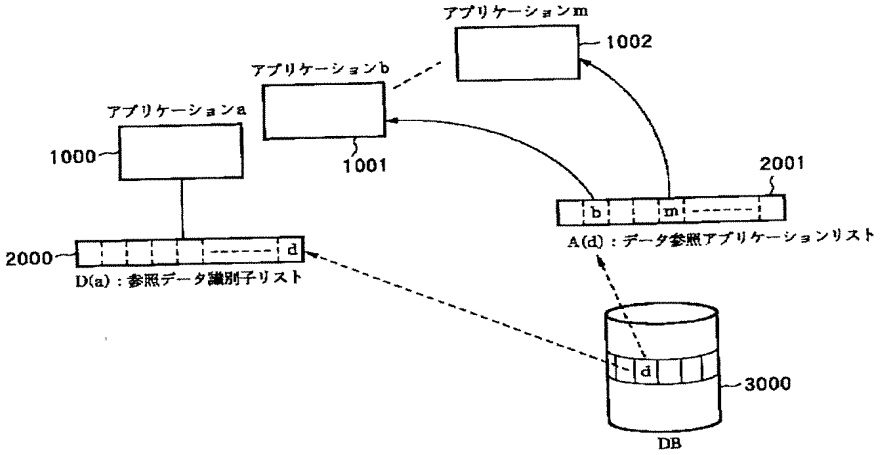
【図７】明示的にロック解放を行う処理手順を示すフローチャートである。

【図８】本実施の形態の情報処理を実行するハードウエア構成の一例を示す図である。

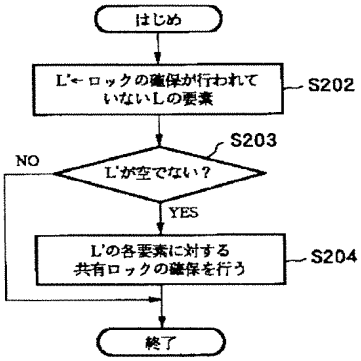
【図1】



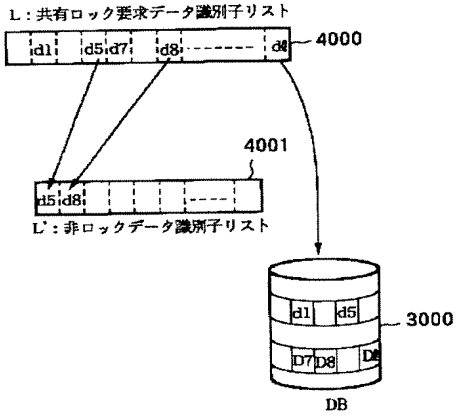
【図2】



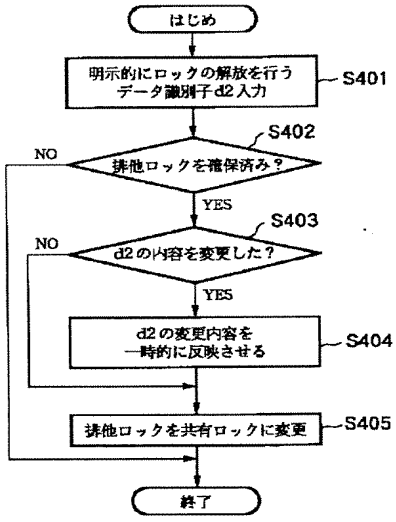
【図3】



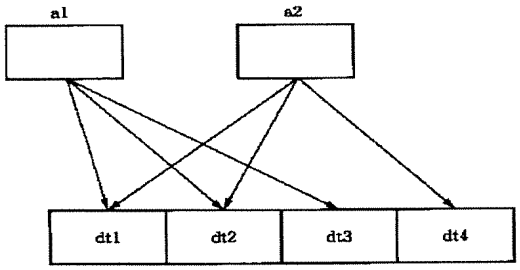
【図4】



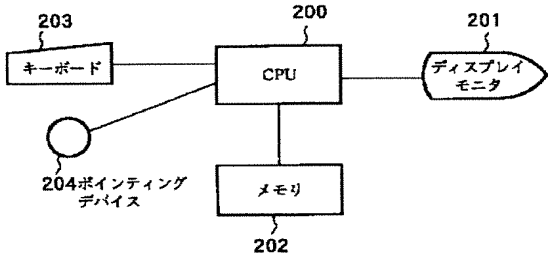
【図7】



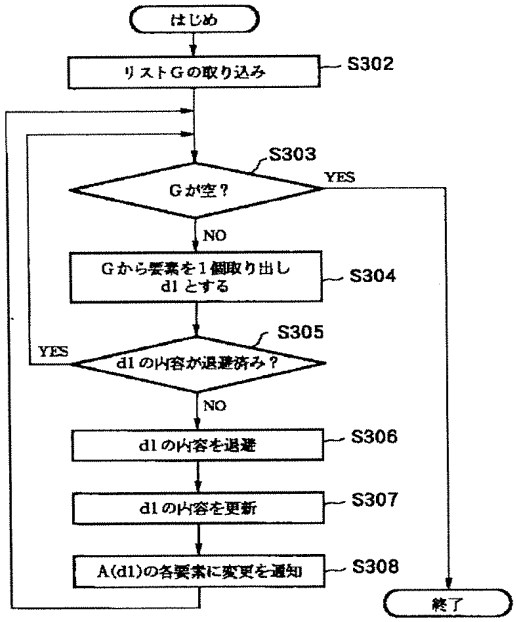
【図6】



【図8】



【図５】


はじめ
リストGの取り込み
S302
S303
Gが空？
YES
NO
Gから要素を１個取り出し
d1 とする
S304
S305
YES
d1の内容が退避済み？
NO
d1の内容を退避
S306
d1の内容を更新
S307
A(d1)の各要素に変更を通知
S308
終了